최소 중복을 이용한 Hotspot 시간 데이터의 관리

윤홍원*·이중화**

Management Strategy of Hotspot Temporal Data using Minimum Overlap

Hong-won Yun*·Jung-hwa Lee**

요 약

과학적 응용에서 발생하는 각종 실험 데이터의 관리에 대한 관심이 높아지고 있으나 체계적인 관리 방법에 대한 연구가 많지 않다. 이 논문에서는 과학적 응용에서 발생하는 군집을 이루는 시간 데이터를 관리하는 방법을 제안한다. 먼저, 시간 데이터를 구분하는 경계값 LB와 RB를 정의하고 과거, 현재, 미래 세그먼트에 각각 저장되는 개체 버전을 정의하였다. 또한 Hotspot 분포를 가지는 시간 데이터에 대하여 각 세그먼트 사이에 이동하는 알고리즘을 나타내었고, 제안하는 최소중복을 이용한 이동 방법과 기존방법에 대하여 성능을 비교하였다. 질의에 대한 평균 응답 시간에서는 기존의 방법과 비슷한 결과를 보였다. 제안한 이동 방법은 세그먼트 사이에 중복해서 저장되는 데이터 수를 적게 하므로 공간 이용율 측면에서는 기존의 이동 방법보다 효율적이었다.

ABSTRACT

We propose a strategy to manage temporal data which are occurred on scientific applications. Firstly, We define LB and RB to separate temporal data, and entity versions to be stored in past, current, future segments. Also, We describe an algorithm to migrate temporal data with hotspot distribution among segments. The performance evaluation of average response time and space utilization is conducted. Average response time between two methods is similar, and space is saved in proposed method.

키워드

Temporal data management, Data migration method

Ⅰ. 서 론

시간 데이터는 이력 데이터와 미래 데이터로 나눌 수 있으며 이력 데이터는 과거 데이터와 현재 데이터를 포함한다. 이력 데이터는 이력이 발생하는 유형에 따라서 변경 이력, 발생 이력, 그리고 진행 이력으로 구분 할 수 있다. 이력 데이터와 미래 데이터를 모두 포함하는 시간 데이터의 보기가 되는 응용은, 비즈니스 응용과 과학적 응용으로 분류할 수 있다. 기존의 데이터 관리 방법들은 일반적인 비즈니스 응용의 데이터를 바탕으로 연구가 이루어졌다[1,2].

본 논문에서는 과학적 응용에서 발생하는 데이터를 관리하는 방법에 대하여 연구하고자 한다. 과학적 응용의 보기가 되는 전형적인 시간 데이터의 특성은 물리 실험을 들 수 있는데, 이러한 실험에 대한 반응으로 나타나는 시간 데이터는 일정한 시간 동안 군집을 이루고 이산적인 특성을 가지고 있다[3,4]. 이 논문에서는 과학적 응용에서 발생하는 시간 데이터의 특성을

* 신라대학교
** 동의대학교

접수일자 : 2005. 5. 27

고려해서 시간적으로 연관성이 많은 개체 버전을 관리하는 방법에 대해서 살펴본다. 먼저, 시간 데이터를 과거, 현재, 그리고 미래 데이터로 구분하는데 사용하는 경계값을 정의한다. 그리고, 각 세그먼트에 저장되는 개체 버전을 정의하고 세그먼트 사이에 데이터를 이동하는 알고리즘을 나타낸다. 제안하는 시간 데이터 이동 방법과 기존의 이동 방법 사이에 성능을 비교하고 평가한다.

Ⅱ. Hotspot 시간 데이터

개체 버전이 군집을 이루는 데이터에 대해서는 개체 버전 사이에 연관성을 고려할 필요가 있다[3,5,6,7]. 군집을 이루는 데이터에서 임의의 시점 질의가 주어질 때 임의의 개체 버전이 질의를 만족하려면 그 개체 버전의 유효 시간 간격이 질의 시점을 포함해야 한다. 임의의 시점 질의에 대해서 임의의 두 개체 버전이 동시에 선택되려면 두 개체 버전의 유효 시간이 겹쳐야 한다. 두 개체 버전의 겹침의 정도와 두 개체 버전이 동시에 선택될 확률은 밀접한 관련이 있다. 직관적으로 두 개체 버전의 유효 시간이 많이 겹칠수록 동시에 선택될 확률이 높음을 알 수 있다.

그림 1 에서 개체 버전 P_1과 P_2가 겹치는 시간을 A_1이라고 하고 P_2와 P_3이 겹치는 시간을 A_2라고 하자. A_1시간이 A_2시간보다 길다고 가정하면 임의의 시점 질의 대해서 P_1과 P_2가 동시에 선택될 확률이 P_2와 P_3가 동시에 선택될 확률보다 큼을 알 수 있다. 임의의 두 개체 버전이 동시에 선택될 시간은 다음과 같이 나타낼 수 있다

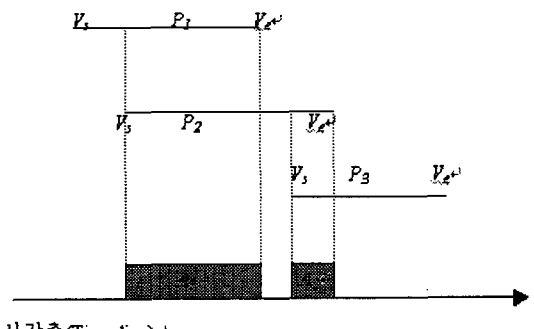


그림 1. 시간질의와 유효시간과의 관계
Fig. 1 Relation between temporal query and valid time

$A(P_1,P_2)=[\max(P_1.V_s,P_2.V_s),\ \min(P_1.V_e,P_2.V_e)]$
if $P_1.V_s \le P_2.V_e \wedge P_1.V_e$

유효 시간 간격이 겹치는 개체 버전들은 동시에 선택될 확률이 높기 때문에 이러한 개체 버전은 같은 세그먼트에 두는 것이 바람직하다.

Ⅲ. 경계값 정의

과거 세그먼트와 현재 세그먼트에 저장되는 개체 버전을 구분하는 기준이 되는 시점을 *LB* (*Left Bound with min overlap*), 현재 세그먼트와 미래 세그먼트에 저장되는 개체 버전을 구분하는 기준이 되는 시점을 *RB* (*Right Bound with min overlap*) 라 하고, 경계값으로 사용되는 *LB*와 *RB*를 정의하도록 한다.

표 1. 세그먼트와 저장되는 개체버전의 집합
Table. 1 Segment and stored entity version sets

세그먼트	저장되는 개체버전 집합
과거 세그먼트	$P=\{Eij \mid Eij.Ve<LB\}$
현재 세그먼트	$C=\{Eij \mid (Eij.Vs \ge LB) \wedge (Eij.Ve<RB)\}$
미래 세그먼트	$F=\{Eij \mid Eij.Vs \ge RB\}$
과거∧현재 세그먼트	$PC=\{Eij \mid (Eij.Vs<LB) \wedge (LB<Eij.Ve<RB)\}$
현재∧미래 세그먼트	$CF=\{Eij \mid (LB \le Eij.VsRLB) \wedge (Eij.Ve>RB)\}$

따라서 과거 세그먼트, 현재 세그먼트, 그리고 미래 세그먼트가 포함하는 모든 개체 버전의 집합은 다음과 같이 정의할 수 있다.

$$EP=P\bigcup_{All}PC\ ,\quad EC=PC\bigcup_{All}C\bigcup_{All}CF,\quad EF=CF\bigcup_{All}F$$

EP, *EC*, 그리고 *EF*에 들어있는 개체 버전의 유효 시간 간격의 집합을 각각 *IP*, *IC*, 그리고 *If*라고 하고 다음과 같이 나타낸다. *Ip={ip1,ip2, ..., ipl}*, *Ic={ic1,ic2, ..., icm}*, *If={if1,if2, ..., ifm}*

이 집합에 들어있는 모든 유효시간 간격의 개수는 다음과 같이 표현한다. $|Ip|=np$, $|Ic|=nc$, $|If|=nf$

임의의 시점t에서 서로 겹치는 시간 간격의 개수를 구하는 함수는 다음과 같다. $OIc(t)=|\{ic \in Ic | ic.Vs<ic.Ve\}|$, $OIf(t)=|\{if \in If | if.Vs<t<if.Ve\}|$, $SIc(t)=|\{ic \in Ic | ic.Vs=t\}|$, $EIc(t)=|\{ic \in Ic | ic.Ve=t\}|$, $SIf(t)=|\{if \in If | if.Vs=t\}|$, $EIf=|\{if \in If | if.Ve=t\}|$

과거세그먼트, 현재 세그먼트, 그리고 미래 세그먼트 각각에 들어가는 시간 간격의 집합에서 임의의 시점 t를 포함하는 시간 범위를 각각 *R(Ip)*, *R(Ic)*, 그리고 *R(If)*라고 하고 다음과 같이 정의한다.

$$R(Ip)=\bigcup_{\{[vsve]\in Ip} [Vs,Ve],\ R(Ic)=\bigcup_{\{[vsve]\in Ic} [Vs,Ve],\ R(If)=\bigcup_{\{[vsve]\in If} [Vs,Ve]$$

R(Ip), *R(Ic)*, 그리고 *R(If)*에 들어가는 유효 시작 시각의 집합을 S(Ip), S(Ic), 그리고 S(If) 로 나타내고, 유효 끝 시각의 집합을 E(Ip), E(Ic), 그리고 E(If) 로 나타내고 이것을 정의하면 다음과 같다.

$S(Ip)=\{Ip.Vs \mid \exists Vs \in R(Ip)$ *such that* $[Vs,Ve] \in Ip\}$
$E(Ip)=\{Ip.Ve \mid \exists Ve \in R(Ip)$ *such that* $[Vs,Ve] \in Ip\}$
$S(Ic)=\{Ic.Vs \mid \exists Vs \in R(Ic)$ *such that* $[Vs,Ve] \in Ic\}$
$E(Ic)=\{Ic.Ve \mid \exists Ve \in R(Ic)$ *such that* $[Vs,Ve] \in Ic\}$
$S(If)=\{If.Vs \mid \exists Vs \in R(If)$ *such that* $[Vs,Ve] \in If\}$
$E(If)=\{If.Ve \mid \exists Ve \in R(If)$ *such that* $[Vs,Ve] \in If\}$

Ic의임의의 시점에서 왼쪽으로 가장 가까운 유효 시작 시각을 *nsleft(Ic.t)*라고 하고, 이것은 다음과 같이 정의한다.

$$nsleft(Ic.t)=max\{Ic.Vs \mid Ic.Vs \in S(Ic)\}$$

*Ic*의임의의 시점부터 왼쪽으로 가장 가까운 유효 시작 시각에서 겹치는 유효 시간 간격의 개수는 *OIc(nsleft(Ic.t))* 로 나타낼 수 있다. 새로운 *LB*를 구하는데 해당하는 시간 영역은, 현재 사용하고 있는 *LB + 1*에서 새 *LB* 를 구하려는 시점 t까지가 된다. 새로운 LB 를 구하는데 해당하는 시간 영역을 *S(Ic)* 라고 하고 다음과 같이 정의한다. $S(Ic) = [\ LB + 1,\ t\ ]$

새로운 *LB*를 구하는 시점을 t라고 하고, *t*에서 왼쪽으로 가장 가까운 유효 시작 시각은 *nsleft(Ic.t)* 이다. *S(Ic)*에서 나오는 모든 유효 시작 시각은 *nsleft (Ic.t)* , *nsleft (nsleft (Ic.t))*, 가 된다. *nsleft(Ic.t)* 를 통해서 나온 바로 앞의 유효 시작 시각을 차례대로 *st1, st2,* 라고 하면, 값의 범위는 $LB + 1 < st1, st2, < t$ 이고, 이 유효 시작 시각 각각의 시점에서 겹치는 시간 간격의 개수는 *OIc(st1), OIc(st2), OIc(st3),* 로 나타낼 수 있다. 이 중에서 가장 최소값이 되는 유효 시작 시각이 과거 세그먼트와 현재 세그먼트를 구분하는 경계값 *LB*가 되고, 이를 정의하면 다음과 같다.

$$LB = min\ \{sti \mid OIc(\ st1\ ),\ OIc(\ st2\ ),\ \ldots\ ,\ OIc(\ stn\ )$$

If의 임의의 시점에서 오른쪽으로 가장 가까운 유효 끝 시각을 *neright (If.t)* 라고 하고, 다음과 같이 정의한다. $neright(If.t)=min\{If.Ve \mid If.Ve \in E(If)\}$

*If*의 임의의 시점부터 오른쪽으로 가장 가까운 유효 끝 시각에서 겹치는 유효 시간 간격의 개수는 *OIf (neright (If.t))*로 나타낼 수 있다. 새로운 *RB* 를 구하는 시점을 *t*라고 하면 새로운 *RB* 를 구하는데 해당되는 시간 영역은, 시점 *t*에서 *LB* 까지의 시간 길이를 *t* 를 중심으로접었을 때 *LB*가 만나는 시점을, *RB* 를 구하기 위한 구간의 끝 시점으로 한다. 시점 *t*에서 *LB* 까지의 길이 1은 t - *LB*가 된다. 새로운 RB 를 구하는데 해당되는 영역은 시점 $t + 1$에서 $t + 1$인데, 이를 *S(If)* 라고 하고 다음과 같이 정의한다. $S(If) = [\ t + 1,\ t + l\ ]$

새로운 *RB* 를 구하는 시점을 *t*라고 하면, $t + 1$에서 오른쪽으로 가장 가까운 유효 끝 시각은*neright (If.t)* 이다. *S(If)*에서 나오는 모든 유효 끝 시각은*neright (If.t)* , *neright (neright(If.t))*, 가 된다. *neright (If.t)* 를 통해서 나온 바로 앞의 유효 끝 시각을 차례대로 *et1, et2,* 라고 하면, 값의 범위는 $t + 1 < et1, et2, < t + l$ 이고, 이 유효 끝 시각 각각의 시점에서 겹치는 시간 간격의 개수는 *OIf (et1), OIf (et2), OIf (et3),* 로 나타낼 수 있다. 이 중에서 가장 최소값이 되는 유효 끝 시각이 현재 세그먼트와 미래 세그먼트를 구분하는 경계값 *RB*가 되고, 이를 정의하면 다음과 같다.

$$RB = min\ \{eti \mid OIf\ (et1),\ OIf\ (et2),\ \ldots\ ,\ OIf\ (etn)\}$$

*LB*를 구하는 알고리즘은 다음과 같다.

```
GetLB() ----------------------------------------------------
begin
/* for all Eij in Future Segment and Current Segment */
 LBpre <- LB;  RBpre <- RB;  LB, RB, TABLE[] <- null;
 while (not eof (EntityVersion <- read (FutSeg))) do
  if ( LBpre < EnitityVersion.Vs < RBpre) then
    Append (TABLE[k++], EnitityVersion.Vs);
  end if
end while
while (not eof (StartTime <- (TABLE[k++], EnitityVersion.Vs)) do
  while (not eof (EntityVersion <- read (FutSeg)) do
    if ( EnitityVersion.Vs < StartTime < EnitityVersion.Ve ) then
    Overlap <- Overlap + 1;
    end if
    Append (TABLE[k], Overlap);
  end while
end while
  while (not eof (EntityVersion <- read (CurSeg))) do
  if ( LBpre < EnitityVersion.Vs < RBpre) then
    Append (TABLE[k++], EnitityVersion.Vs);
  end if
end while
while (not eof (StartTime <- (TABLE[k++], EnitityVersion.Vs)) do
  while (not eof (EntityVersion <- read (CurSeg)) do
    if ( EnitityVersion.Vs < StartTime < EnitityVersion.Ve ) then
    Overlap <- Overlap + 1;
    end if
    Append (TABLE[k], Overlap);
end while
LB <- min{EnitityVersion.Vs | (TABLE[k], Overlap)};
return LB
end ----------------------------------------------------
```

*RB*를 구하는 알고리즘은 다음과 같다.

```
GetRB() ----------------------------------------------------
begin
/* for all Eij in Future Segment and Current Segment */
RBpre <- RB; l <- now-LB; Fin <- now+l; LB, RB, TABLE[] <- null;
 while (not eof (EntityVersion <- read (FutSeg))) do
  if ( RBpre < EnitityVersion.Ve < Fin) then
    Append (TABLE[k++], EnitityVersion.Ve);
  end if
end while
while (not eof (EndTime <- (TABLE[k++], EnitityVersion.Ve)) do
  while (not eof (EntityVersion <- read (FutSeg)) do
    if ( EnitityVersion.Vs < EndTime < EnitityVersion.Ve ) then
    Overlap <- Overlap + 1;
    end if
    Append (TABLE[k], Overlap);
  end while
end while
  while (not eof (EntityVersion <- read (CurSeg))) do
  if ( RBpre < EnitityVersion.Ve < Fin) then
    Append (TABLE[k++], EnitityVersion.Ve);
  end if
end while
while (not eof (EndTime <- (TABLE[k++], EnitityVersion.Vd)) do
  while (not eof (EntityVersion <- read (CurSeg)) do
    if ( EnitityVersion.Vs < EndTime < EnitityVersion.Ve ) then
    Overlap <- Overlap + 1;
    end if
    Append (TABLE[k], Overlap);
end while
RB <- min{EnitityVersion.Ve | (TABLE[k], Overlap)};
return RB
end ----------------------------------------------------
```

표 2. 이동 및 복사 대상 개체버전
Table. 2 Moving and copying entities

작업	세그먼트	해당 개체버전
이동	미래→현재	$(Eij.Vs \geq LB) \wedge (Eij.Ve < RB)$
	현재→과거	$Eij.Ve < LB$
복사	미래→현재	$(LB \leq Eij.Vs < RB) \wedge (Eij.Ve > RB)$
	현재→과거	$(Eij.Vs < LB) \wedge (Eij.Ve < RB)$

데이터를 이동하는 과정에서 발생하는 작업은 이동과 복사로 나눌 수 있다. 표 2는 최소중복에 의한 데이터 이동 방법에서 데이터를 이동할 때 수행하는 이동, 복사의 대상이 되는 개체 버전을 나타내고 있다.

Ⅳ. 성능평가

성능 평가 모델은 질의 생성기, 질의 처리기, 데이터 이동 프로세서로 구성된다. 질의 생성기는 시점 질의와 범위 질의를 지수 분포에 의해서 만들어내고 만들어진 질의는 질의 대기 큐에 들어간다. 질의 처리기는 질의 대기 큐에서 하나의 질의를 꺼내어 시점 질의인가, 범위 질의인가를 구분하고 검색할 세그먼트를 결정한 뒤에 해당 세그먼트를 읽는다.

데이터 이동 프로세서 는 데이터 이동 방법에 따라 현재 세그먼트와 미래 세그먼트를 검사해서 다른 세그먼트로 옮기거나 복사한다. 데이터 이동 프로세서가 현재 세그먼트와 미래 세그먼트를 검사하고 이동하는

중에 발생한 모든 질의는 데이터 이동이 완료될 때까지 질의 대기 큐에서 기다리고 데이터 이동이 끝나면 질의를 계속 처리한다.

최소 중복에 의한 이동 방법은 일정한 시간 동안 군집을 이루는 시간 데이터에 적합한 이동 방법이므로, 모의 실험 데이터는 주기적으로 군집을 이루는 핫스팟 분포(hotspot distribution)를 이용해서 생성하였다. 하나의 군집에는 1% 내지 2% 정도 튜플이 들어가도록 하였고, LLT는 1% 와 5% 로 각각 만들었고 튜플의 길이는 균등 분포로 생성하였다. 이 모의 실험 데이터는 LLT의 비율을 1% 와 5% 로 각각 생성하였다

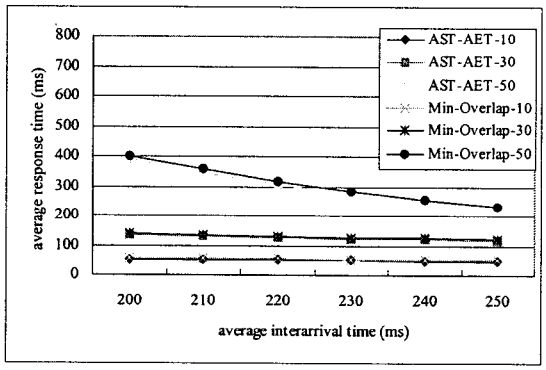


그림 2. 평균응답시간
Fig. 2 Average response time

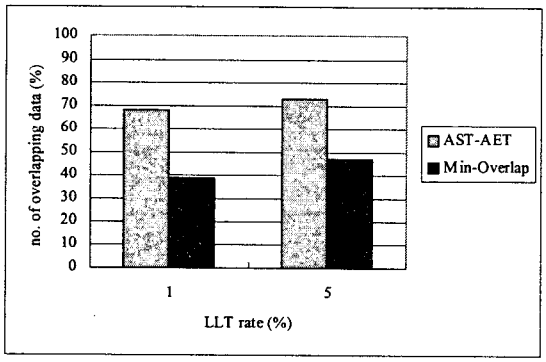


그림 3. 중복 저장되는 데이터의 수
Fig. 3 Number of overlapping data

그림 2는 분리 저장 구조에서 LLT가 1% 이고 시간 질의가 10%, 30%, 그리고 50% 인 경우에 평균에 의한 이동 방법과 최소 중복에 의한 이동 방법에 대해서 사용자 질의에 대한 응답 시간을 측정한 것이다. 두 가지 이동 방법은 시간 질의의 비율이 같은 경우에 사용자 질의에 대한 응답 시간이 거의 비슷하게 나타나고 있다. 이것은 평균에 의한 이동 방법과 최소 중복에 의한 이동 방법은 현재 세그먼트의 크기가 비슷하고 데이터 이동 빈도가 비슷하기 때문이다.

사용자 질의에 대한 성능 측면에서 보면, 평균에 의한 이동 방법과 최소 중복에 의한 이동 방법에서 비슷한 성능을 보이지만, 최소 중복에 의한 방법은 두 세그먼트에 중복되는 개체 버전의 수를 작게 하므로 공간 이용율의 측면에서 효율적이다. 그림3은 현재 세그먼트에 대해서 중복 저장되는 개체 버전의 수를 백분율로 나타낸 것이다. 최소 중복에 의한 이동 방법은 군집을 이루는 실세계의 데이터를 한 세그먼트에 저장함으로써 서로 시간적으로 연관성이 있는 개체 버전을 묶을 수 있기 때문에 실세계의 특징을 그대로 반영하는 특징을 가지고 있다

V. 결 론

과학적 응용에서 발생하는 군집을 이루는 시간 데이터에 적합한 이동 방법으로써 최소 중복에 의한 이동 방법을 제안하였다. 최소 중복에 의한 이동 방법에서 시간 데이터를 구분하는 경계값 LB와 RB를 정의하고 각 세그먼트에 저장되는 개체 버전을 정의하였다. LB와 RB를 구하는 방법을 보이고, 세그먼트 사이에 데이터를 알고리즘을 보였다. 과학적 응용에서 발생하는 군집을 이루는 시간 데이터에 적합한 이동 방법으로 제안한 최소 중복에 의한 이동 방법은, 질의에 대한 평균 응답 시간에서 평균에 의한 이동 방법과 비슷한 결과를 보였다. 최소 중복에 의한 이동 방법은 세그먼트 사이에 중복해서 저장되는 데이터 수를 적게 하므로 공간 이용율 측면에서는 평균에 의한 이동 방법보다 효율적이었다.

참고문헌

[1] I. Ahn and R.T. Snodgrass, "Partitioned Storage for Temporal Databases," Information Systems, vol. 13, no.

4, pp. 369-391, 1986.

[2] I. Ahn and R.T. Snodgrass, "Performance Evaluation of a Temporal Database Management System," Proceedings of the ACM International Conference on Management of Data, pp. 96-107, May 1988.

[3] G. Ozsoyoglu and R.T. Snodgrass, "Temporal and Real-Time Databases: A Survey," IEEE Transactions on Knowledge and Data Engineering, vol. 7, no. 4, pp. 511-532, August 1995.

[4] M.H. Bohlen, "Temporal Database System Implementation", ACM SIGMOD Record, vol. 24, no. 4, pp.53-60, December 1995.

[5] C.S. Jensen, R. Elmasri, S.K. Gadia, P. Hayes, and S. Jajodia(eds), "A Glossary of Temporal Database Concepts," ACM SIGMOD Record, vol. 23, no. 1, pp. 52-64, March 1994.

[6] 박 유현, 이 중화, 윤 홍원, 김 경석, "분리 저장 구조를 가지는 시간지원 데이터베이스에서 데이터 이동 기법에 관한 연구," 한국정보과학회, '97가을학술발표논문집, 24권 2호, pp. 469-472, 1997, 10.

[7] 윤 홍원, 김 경석, "시간지원 데이터베이스에서 자료의 이동을 고려한 저장 방법의 성능 평가," 한국정보과학회논문지, 25권 5호, pp. 756-767, 1998.5.

저자소개

윤홍원(Hong-won Yun)

E-mail: hwyun@silla.ac.kr

1986년 부산대학교 계산통계학과 졸업(학사)

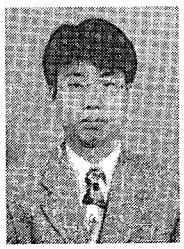
1990년 한국외국어대학교 경영정보대학원 전자계산학과(이학석사)

1998년 부산대학교 대학원 전자계산학과(이학박사)

2003년 North Carolina State University 객원교수

1996년 ~ 현재 신라대학교(구.부산여자대학교) 컴퓨터정보공학부 부교수

※관심분야: 데이터베이스, 시간 데이터베이스, 시맨틱 웹, e-러닝



이중화(Jung-hwa Lee)

E-mail: junghwa@deu.ac.kr

1988.3-1992.2 부산대학교 전자계산학과(이학사)

1993.3-1995.2 부산대학교 전자계산학과(이학석사)

1995.3-2001.8 부산대학교 전자계산학과(이학박사)

2002.3 - 현재 동의대학교 컴퓨터-소프트웨어공학부 조교수

※관심분야: 데이터베이스, XML, 시맨틱 웹